Midterm report: Unilever NLP

ENGIE4800 **-** Data Science Capstone
Ian Johnson (icj2103) & Julien Maudet (jm4418)
GitHub: https://github.com/Unilever-NLP/unilever-nlp

I. Introduction (jm)

In this project, which is a collaboration between Columbia University and Unilever, we aim at developing a web interface that would enable analysts at Unilever extract meaningful information from textual survey data that they gather from customers around the globe. If time permits, the interface will also be able to process amazon reviews on Unilever beauty products.

The web interface will perform two main tasks: a summarization of all answers to a given question and the extraction of the most relevant keyphrases in the answers. In this report, we will start with introducing the datasets then move on to presenting both tasks and the web interface.

II. Data sources (jm)

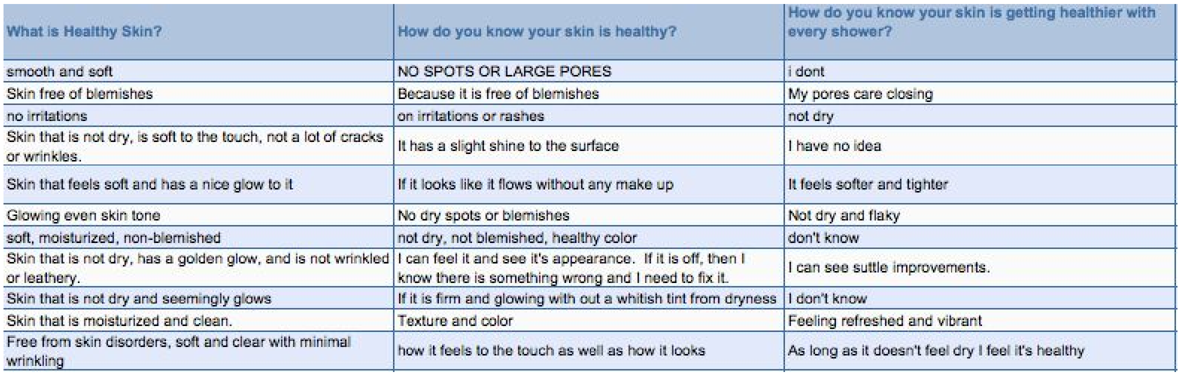
We work with two types of datasets here, amazon reviews on beauty products and survey data.

II.1 Survey data

It consists in 198, 468 survey answers to the following questions:

- What is healthy skin?
- How do you know your skin is healthy?
- How do you know your skin is getting healthier with every shower?
- How do you get healthy skin?
- How do you maintain healthy skin?
- How bar soap or body wash gives you healthy skin?
- How concerned?

Here's a snapshot of the dataset:

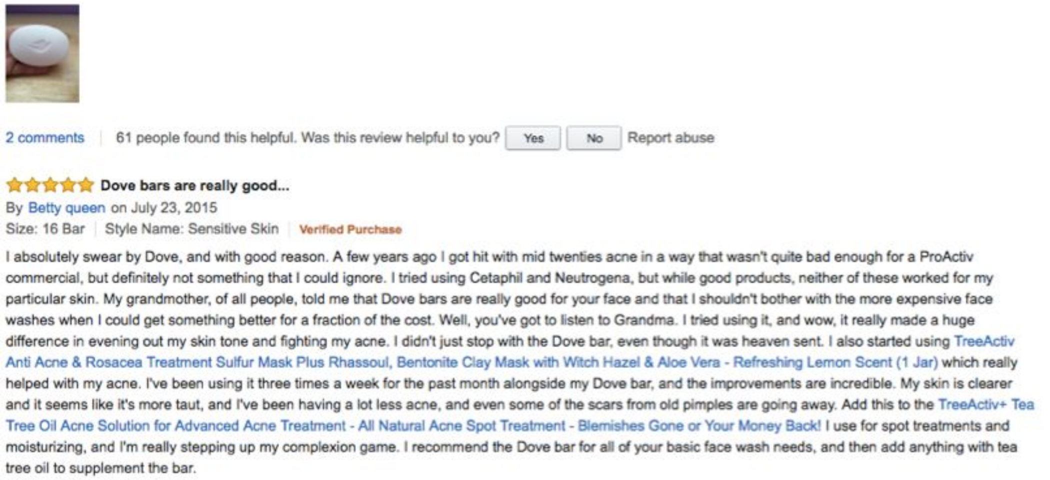


What is Healthy Skin?	How do you know your skin is healthy?	How do you know your skin is getting healthier with every shower?
smooth and soft	NO SPOTS OR LARGE PORES	i dont
Skin free of blemishes	Because it is free of blemishes	My pores care closing
no irritations	on irritations or rashes	not dry
Skin that is not dry, is soft to the touch, not a lot of cracks or wrinkles.	It has a slight shine to the surface	I have no idea
Skin that feels soft and has a nice glow to it	If it looks like it flows without any make up	It feels softer and tighter
Glowing even skin tone	No dry spots or blemishes	Not dry and flaky
soft, moisturized, non-blemished	not dry, not blemished, healthy color	don't know
Skin that is not dry, has a golden glow, and is not wrinkled or leathery.	I can feel it and see it's appearance. If it is off, then I know there is something wrong and I need to fix it.	I can see suttle improvements.
Skin that is not dry and seemingly glows	If it is firm and glowing with out a whitish tint from dryness	I don't know
Skin that is moisturized and clean.	Texture and color	Feeling refreshed and vibrant
Free from skin disorders, soft and clear with minimal wrinkling	how it feels to the touch as well as how it looks	As long as it doesn't feel dry I feel it's healthy

We process the data by concatenating all answers to a given question into a single text. It basically gives us one large character chain per question that we can perform NLP analysis on.

II.2 Amazon reviews

The whole dataset contains 198, 502 Amazon reviews that originally look like the following:



61 people found this helpful. Was this review helpful to you?

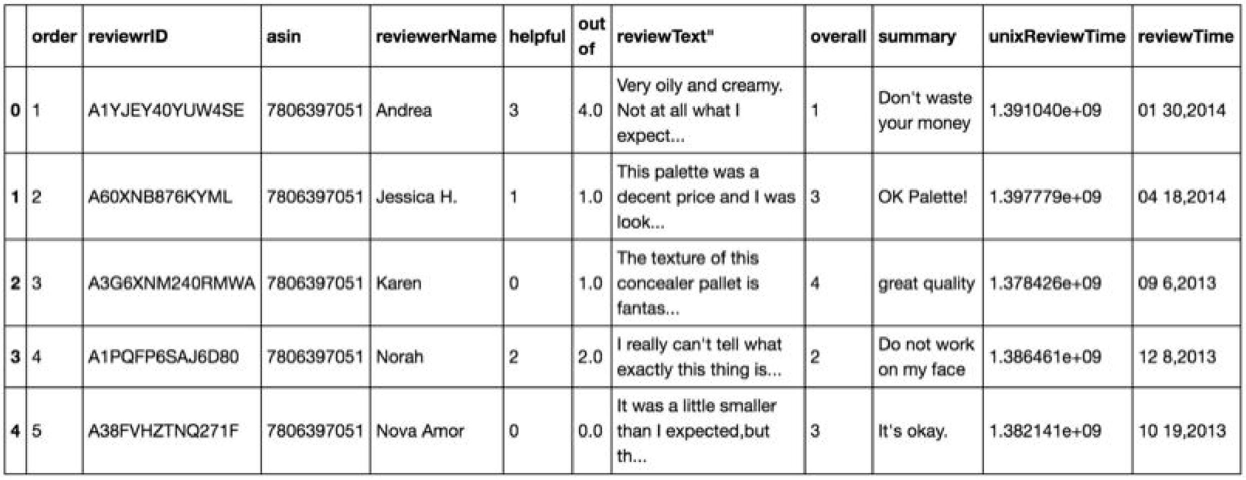
★★★★★ **Dove bars are really good...**

By Betty queen on July 23, 2015

Size: 16 Bar | Style Name: Sensitive Skin | Verified Purchase

I absolutely swear by Dove, and with good reason. A few years ago I got hit with mid twenties acne in a way that wasn't quite bad enough for a ProActiv commercial, but definitely not something that I could ignore. I tried using Cetaphil and Neutrogena, but while good products, neither of these worked for my particular skin. My grandmother, of all people, told me that Dove bars are really good for your face and that I shouldn't bother with the more expensive face washes when I could get something better for a fraction of the cost. Well, you've got to listen to Grandma. I tried using it, and wow, it really made a huge difference in evening out my skin tone and fighting my acne. I didn't just stop with the Dove bar, even though it was heaven sent. I also started using TreeActiv Anti Acne & Rosacea Treatment Sulfur Mask Plus Rhassoul, Bentonite Clay Mask with Witch Hazel & Aloe Vera - Refreshing Lemon Scent (1 Jar) which really helped with my acne. I've been using it three times a week for the past month alongside my Dove bar, and the improvements are incredible. My skin is clearer and it seems like it's more taut, and I've been having a lot less acne, and even some of the scars from old pimples are going away. Add this to the TreeActiv+ Tea Tree Oil Acne Solution for Advanced Acne Treatment - All Natural Acne Spot Treatment - Blemishes Gone or Your Money Back! I use for spot treatments and moisturizing, and I'm really stepping up my complexion game. I recommend the Dove bar for all of your basic face wash needs, and then add anything with tea tree oil to supplement the bar.

They are already preprocessed and here is the format of the dataset as we will use it:



	order	reviewrID	asin	reviewerName	helpful	out of	reviewText"	overall	summary	unixReviewTime	reviewTime
0	1	A1YJEY40YUW4SE	7806397051	Andrea	3	4.0	Very oily and creamy. Not at all what I expect...	1	Don't waste your money	1.391040e+09	01 30,2014
1	2	A60XNB876KYML	7806397051	Jessica H.	1	1.0	This palette was a decent price and I was look...	3	OK Palette!	1.397779e+09	04 18,2014
2	3	A3G6XNM240RMWA	7806397051	Karen	0	1.0	The texture of this concealer pallet is fantas...	4	great quality	1.378426e+09	09 6,2013
3	4	A1PQFP6SAJ6D80	7806397051	Norah	2	2.0	I really can't tell what exactly this thing is...	2	Do not work on my face	1.386461e+09	12 8,2013
4	5	A38FVHZTNQ271F	7806397051	Nova Amor	0	0.0	It was a little smaller than I expected,but th...	3	It's okay.	1.382141e+09	10 19,2013

We haven’t included the possibility to perform analysis on Amazon reviews in the interface yet, but this is on the agenda.

III. Keyphrase Extraction (jm)

Extracting keyphrases from a long document enables a very quick insight in the content of the text. In our case, the analyst can easily understand the different concerns of customers, without having to go through all survey answers.

Keyphrase extraction is an entire research area in Natural Language Processing and many algorithms exist. The two most commonly used are TextRank [2] and RAKE [1]. Based on the research paper, RAKE achieves better or similar performance on the task itself while being much more computationally efficient. This led us to consider this algorithm first. Furthermore, it is open source (MIT License) and can thus be used at Unilever.

III.1 Description of the algorithm

It is important to note that the algorithm is unsupervised and can thus be adapted to every language. It matters in our case as it could process surveys conducted among Unilever customers from any country.

Input of the algorithm:
- document
- list of stop words and phrase delimiters (the, a, from…)
- parameters:
  - minimum length of a word in a keyphrase
  - minimum frequency for a word in the text
  - maximum number of words per keyphrase

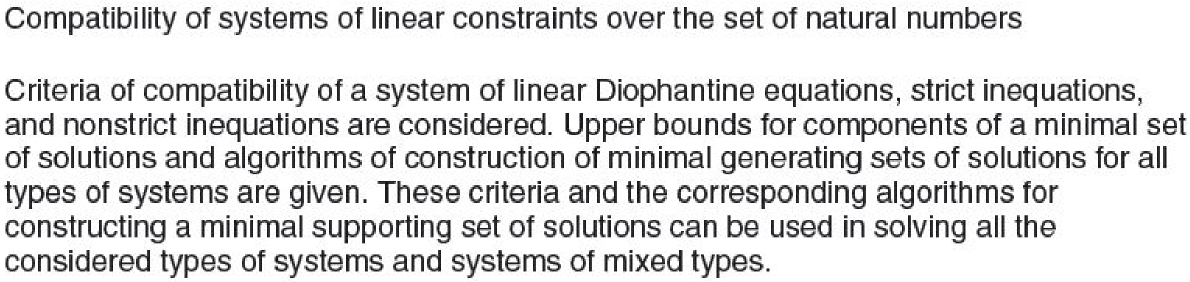
Output:
- List of keyphrases and the associated relevance score

The algorithm starts by parsing the text based on the given parameters and the list of stop words and phrase delimiters. This step results in a set of candidate keyphrases.
Then, a relevance score is computed for each keyphrase, which is the sum of the scores for each word in the keyphrase.

The score of a word in a keyphrase equals the ratio between the degree of the word and the frequency of the word. The degree of a word w is the number of words that appear in keyphrases where w is. The frequency of a word w equals the number of times w appears in the text.
We can then rank all keyphrases based on the relevance scores.

The different steps are illustrated below:

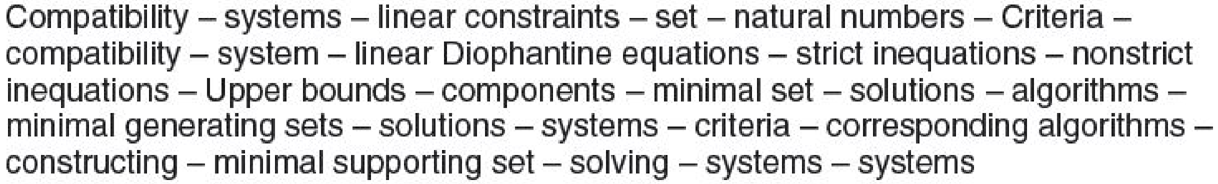
Original text:



Compatibility of systems of linear constraints over the set of natural numbers

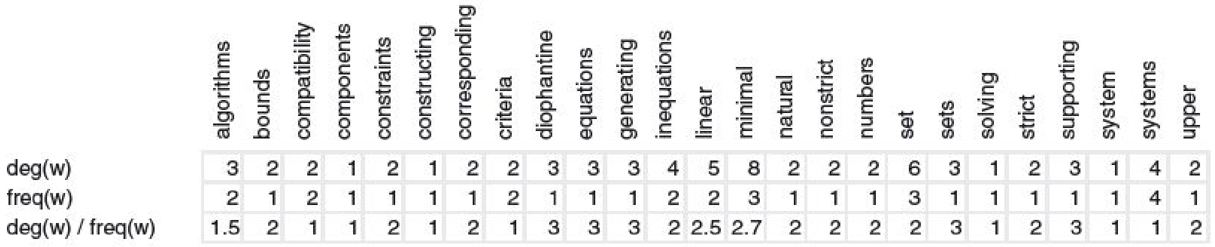
Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types.

Candidate keyphrases:



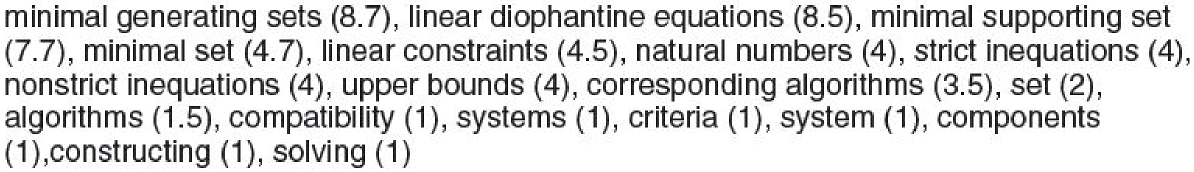
Compatibility – systems – linear constraints – set – natural numbers – Criteria – compatibility – system – linear Diophantine equations – strict inequations – nonstrict inequations – Upper bounds – components – minimal set – solutions – algorithms – minimal generating sets – solutions – systems – criteria – corresponding algorithms – constructing – minimal supporting set – solving – systems – systems

Score for each word:



	algorithms	bounds	compatibility	components	constraints	constructing	corresponding	criteria	diophantine	equations	generating	inequations	linear	minimal	natural	nonstrict	numbers	set	sets	solving	strict	supporting	system	systems	upper
deg(w)	3	2	2	1	2	1	2	2	3	3	3	4	5	8	2	2	2	6	3	1	2	3	1	4	2
freq(w)	2	1	2	1	1	1	1	2	1	1	1	2	2	3	1	1	1	3	1	1	1	1	1	4	1
deg(w) / freq(w)	1.5	2	1	1	2	1	2	1	3	3	3	2	2.5	2.7	2	2	2	2	3	1	2	3	1	1	2

Candidate keyphrases and the associated scores:

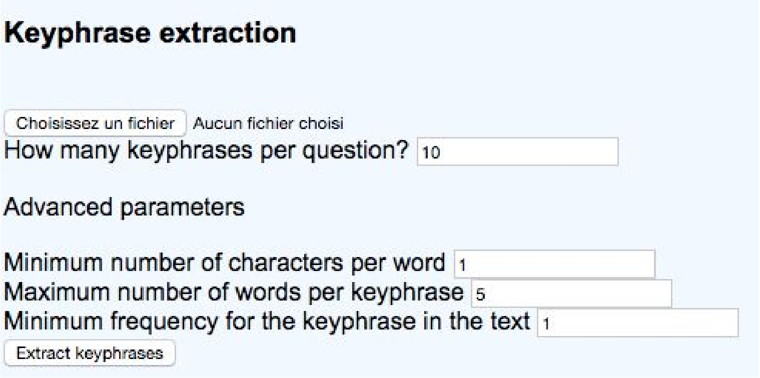


minimal generating sets (8.7), linear diophantine equations (8.5), minimal supporting set (7.7), minimal set (4.7), linear constraints (4.5), natural numbers (4), strict inequations (4), nonstrict inequations (4), upper bounds (4), corresponding algorithms (3.5), set (2), algorithms (1.5), compatibility (1), systems (1), criteria (1), system (1), components (1),constructing (1), solving (1)

IV. Web Interface (jm)

We build the web interface using Python and the Flask package for the back end part, and HTML, CSS and Javascript for the front end.

IV.2 Keyphrase Extraction in the interface



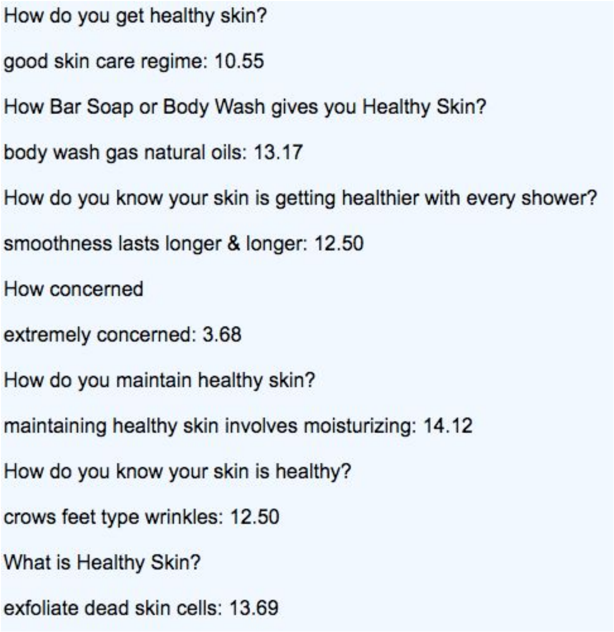
Keyphrase extraction

How many keyphrases per question? 10

Advanced parameters

Minimum number of characters per word 1
Maximum number of words per keyphrase 5
Minimum frequency for the keyphrase in the text 1

After uploading the file to be processed, the analyst can select the number of keyphrases per question, and the three parsing parameters as described earlier. When clicking ‘Extract keyphrases’, the browser send these information to the server which runs the keyphrase extraction algorithm and returns the desired number of keyphrases per question. The browser then displays the answers as follows. In this picture, we put the number of keyphrases to one in order to display all questions:



How do you get healthy skin?

good skin care regime: 10.55

How Bar Soap or Body Wash gives you Healthy Skin?

body wash gas natural oils: 13.17

How do you know your skin is getting healthier with every shower?

smoothness lasts longer & longer: 12.50

How concerned

extremely concerned: 3.68

How do you maintain healthy skin?

maintaining healthy skin involves moisturizing: 14.12

How do you know your skin is healthy?

crows feet type wrinkles: 12.50

What is Healthy Skin?

exfoliate dead skin cells: 13.69

V. Coming next (jm)

V.1 Keyphrase Extraction

The next step in this task is to improve the algorithm by refining the set of candidate keyphrases using synonyms and taking care of spelling errors in the answers. Indeed, some extracted keyphrases are redundant like ‘feels soft’ and ‘feels softer’. Another approach to be implemented in order to refine the set of candidate keyphrases one step further is to use word2vec and not only synonyms.

VI. Text Summarization (ij)

In this section we present our initial work towards implementing the Semantic Volume Maximization [3] algorithm for text summarization on the survey responses dataset.

V.1 Preprocessing

The first step of the summarizer pipeline is to make sentences from reading in the dataframe using the Nltk sentence tokenizer. After this step, we include the option to split longer sentences with a parameter designating the number of words at which to split. This is a preliminary step towards solving the issue of longer sentences being more likely to appear in the final summary, as their distance from the centroid is larger, and thus makes it more likely for these sentences to have a larger distance from their projections onto the basis vector subspace. We might benefit from exploring the notion of normalizing all sentence vectors to unit length beforehands, to avoid this issue. In addition, we will explore the idea of splitting sentences based on conjunctions and commas or other types of delimiters, to keep sentences to within a certain length of one another.

At this point in the pipeline, we had previously attempted to spell check each sentence and correct grammatical and spelling errors using the TextBlob package, but this tool was too slow to be run against a large corpus, and created a bottleneck, so we have to to be able to integrate this feature, and may explore other spelling correction options to increase readability of user-generated data.

The next pipeline step was to stem or lemmatize each of the sentences. These methods served the purpose of cutting down words so that they would be considered the same when they had different endings. For instance if a sentence had the word “smooth” and another sentence had the word “smoother” then these would be chopped down to just “smooth” and we would be less likely to choose both of these sentences in our final set, as they would be detected as both representing more similar concepts. At first stemming seemed like the best method for this type of normalization, however it has a tendency to chop unrelated words to the same stem, which would be incorrectly assigning a similar vector to two sentences with previously dissimilar words. Lemmatization on the other hand uses a vocabulary and morphological analysis of the words to more accurately convert the words to their proper lemma, as opposed to simply chopping off the last characters. This provides a more suitable method for maintaining the correct distances between the basis vectors.

The next step used in Yogotama et al. is to remove all bigrams from the vector representations that contain only stopwords. This is a cleaning step that is meant to reduce the noise in the vector representations that is due to words of little relevance to our topics.

The next step in preprocessing is to perform the vectorization of the sentences. Yogotama et al. perform this vectorization using bigram counts (e.g. each dimension in the vector represents the number of occurrences of that particular pair of words in the sentence). This bigram count representation can be achieved using the FeatureHasher from Scikit Learn. We also include the option to perform vectorization based solely on count of individual words, using Scikit Learn's CountVectorizer. We use scipy sparse matrices to preserve space and for efficient computations in later steps.

The final, and most interesting step of preprocessing involves the use of the SVD to represent the term counts in a more condensed form. SVD is used in latent semantic analysis and latent semantic indexing as a method to extract out representations of "topics" from our vectorized documents. These topics can be thought of as groups of words that are often found together in the same documents. This method does not give us a title for these related words - that we have to interpret for ourselves. For example, one topic might consist of the extracted keywords "car" and "train", but we would have to interpret this group of words for ourselves as being related to some sort of topic associated with "transportation".

The SVD method for LSA works by decomposing the original term-document matrix into a set of three new matrices U, Σ, and V^T. The U matrix represents the strength of the relationship between each topic and each term or word, and thus takes the shape *m x r*, with r representing the number of topics. The Sigma matrix is a diagonal matrix of the singular values, each of which is equal to the square root of the eigenvalues of the matrix A^TA, and each of which represents the relative importance of each topic. The V matrix represents the strength of the relationship between each document and each topic, and thus takes the shape *r x n*. One use of the SVD for our purposes, is to decompose the original term-document (or in our case sentence-document) matrix into a more compressed form. We achieve this by selecting only the top k topics in our use of the algorithm (provided by scipy.sparse.linalg.svds). This method preserves the most important semantic information from the sentences, while reducing noise and other artifacts of the original vectorization.

After these preprocessing steps we have our reduced *(# sentences) x (# topics)* matrix which can be used by the text summarization algorithm.

V.1 Semantic Volume Maximization

Given a list of sentences (S_1 … S_n), with their associated vectorized representations (U_1 … U_n), as well as the maximum length of the summary (L), Semantic Volume Maximization returns the set of sentences whose corresponding vectors span a greedily-computed maximal volume of the semantic space. The steps of the algorithm are as follows:

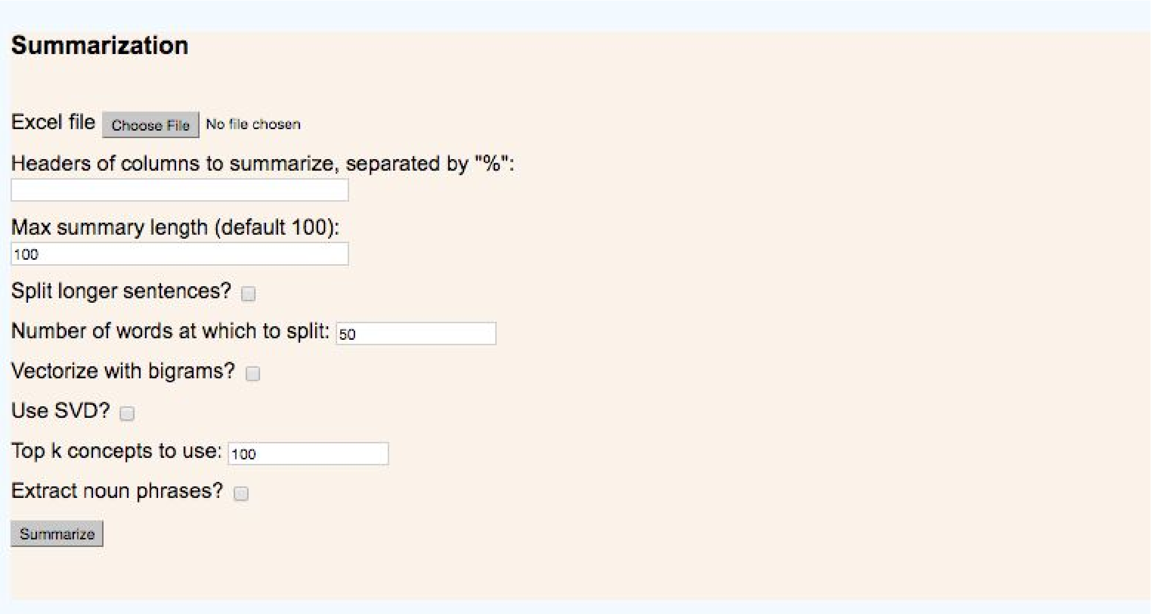
1) Compute the cluster centroid C, the mean basis vector across all vectorized sentences
2) Find the vector U_p most distant from C
3) Find the vector U_q most distant from U_p
4) Compute B_0, the unit vector corresponding to the vector U_q
5) Add B0 to the set B, which will store all further basis vectors
6) Add S_q to the final set of sentences S

7) While the total word length of the sentences in S is less than our limit L:
   a) Compute the vector U_r most distant from the subspace spanned by vectors in B
   b) Add this sentence to S, and add its corresponding normalized vector representation to B
8) Return the list of sentences S

The way that this algorithm works is by iteratively choosing the vector whose projection onto the existing subspace of bases is maximally distant from the vector itself. The authors of the original paper mention that this method can be thought of as finding a convex hull of the subspace of the sentence vectors, however we have a question to whether this is a proper interpretation. In either case, because of the iterative way in which vectors are added to the set B, Semantic Volume Maximization is a greedy algorithm. It may be possible to get stuck in a local maximum of the semantic volume, but we need to pursue further research to verify this claim.

This summarization method, with additional options and parameters can be found in its unfinished state at https://unilever-nlp.herokuapp.com/index.html. An additional parameter can be used to extract noun phrases from the uploaded contents of the excel file. Upcoming work includes the following tasks:

1) Integrating cosine similarity as a distance metric in place of the distance from the projection onto the basis subspace
2) Including an option to use tfidf weighting as opposed to regular word or bigram counts
3) Separate sentences based on conjunctions and delimiters using Stanford's core NLP parser and extracting S tags.
4) Utilizing word embeddings for sentence vectorization values
5) Testing on Amazon product reviews dataset for each product
6) Allowing different file formats and removing requirement to include excel column headers



Summarization

Excel file No file chosen

Headers of columns to summarize, separated by "%":

Max summary length (default 100): 100

Split longer sentences? ☐

Number of words at which to split: 50

Vectorize with bigrams? ☐

Use SVD? ☐

Top k concepts to use: 100

Extract noun phrases? ☐

Text summarization portion of the application interface

VII. Conclusion (ij)

We present here our progress towards a tool for useful analysis of Unilever customer reviews and survey data, which includes keywords extraction and text summarization. Our upcoming work includes improvements to the functionality and usability of these features, as well as the incorporation of new features as suggested by our mentors. We hope that this tool will prove useful in analysis of future reviews and survey datasets.

VI. References

1. Rose, Stuart, et al. "Automatic keyword extraction from individual documents." *Text Mining* (2010): 1-20
2. Mihalcea, Rada, and Paul Tarau. "TextRank: Bringing order into texts." Association for Computational Linguistics, 2004.
3. Yogatama D, Liu F, Smith NA. “Extractive Summarization by Maximizing Semantic Volume.” *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing:* 1961-1966. September, 2015
4. Steinberger, Josef, and Karel Ježek. "Text summarization and singular value decomposition." International Conference on Advances in Information Systems. Springer Berlin Heidelberg, 2004.
5. Joshua Safyan, Shengzhong Yin, Junhui Liao, Shuni Fang. *“Keyword Analysis and Automatic Summarization”*. Fall 2016 ENGIE4800 course, Columbia University.